Eurographics Symposium on Geometry Processing 2021
K. Crane and J. Digne
(Guest Editors)

Volume 40 (2021), Number 5

Normal-Driven Spherical Shape Analogies

Hsueh-Ti Derek Liu and Alec Jacobson
University of Toronoto

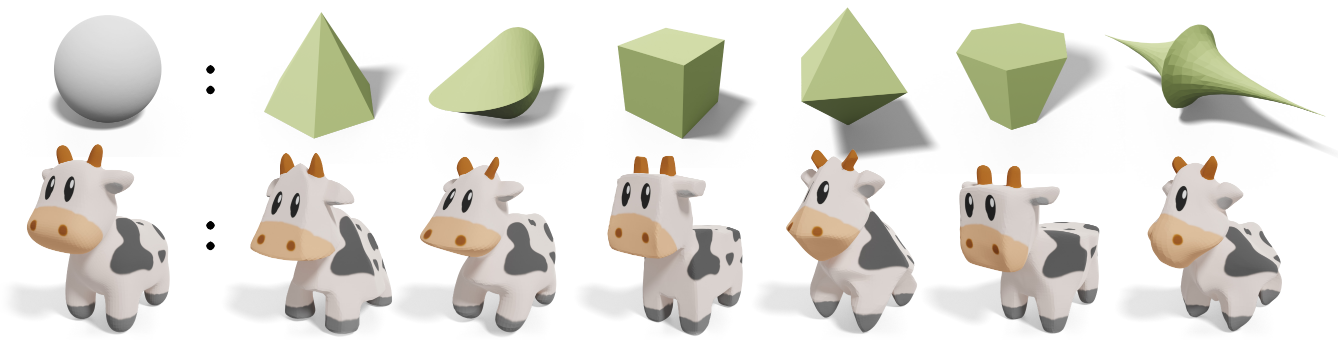


Figure 1: *Our* normal-driven spherical shape analogy *stylizes an input 3D shape (bottom left) by studying how the surface normal of a style shape (green) relates to the surface normal of a sphere (gray).*

Abstract

This paper introduces a new method to stylize 3D geometry. The key observation is that the surface normal is an effective instrument to capture different geometric styles. Centered around this observation, we cast stylization as a shape analogy problem, where the analogy relationship is defined on the surface normal. This formulation can deform a 3D shape into different styles within a single framework. One can plug-and-play different target styles by providing an exemplar shape or an energy-based style description (e.g., developable surfaces). Our surface stylization methodology enables Normal Captures as a geometric counterpart to material captures (MatCaps) used in rendering, and the prototypical concept of Spherical Shape Analogies as a geometric counterpart to image analogies in image processing.

1. Introduction

Analogies of the form $A : A' :: B : B'$ is a reasoning process that conveys *A is to A' as B is to B'*. This formulation has become a core technique for creating artistic 2D digital content, such as image analogies [HJO*01] in Photoshop [Ado21] for image stylization and the Lit Sphere [SMGG01] (a.k.a. MatCap) in ZBrush [Pix20] for non-photorealistic renderings. However, leveraging analogies to stylize 3D geometry is still at a preliminary stage because defining the analogy relationship on surface meshes requires dealing with irregular discretizations, curved metrics, and different topologies.

In this paper, we introduce a step towards a more general 3D shape analogies, named *spherical shape analogies*. We consider a specific case where A is a unit sphere. This restriction enables us to operate on an input mesh B with arbitrary topologies, boundaries, and geometric complexity. While not fully general, because A is restricted to be a sphere, we demonstrate that this formulation can immediately achieve different geometric styles within a single framework. In Fig. 1, we show that by providing different target style shapes A' to the algorithm, we can turn the input shape B into different styles. In addition to stylization, our method can encompass many existing applications, such as developable surface approximation and PolyCube deformation.

One key observation in our spherical shape analogies is that the surface normal is an effective instrument to capture geometric styles. Thus, we define the analogy relationship based on normals: we optimize a stylized shape B' such that the relationship between the surface normals of B and B' is the same as the relationship between the surface normals of A and A'

We realize this by casting it as a simple and effective normal-driven shape optimization problem which aims at deforming the input shape towards a set of desired normals. However, such an optimization problem is often difficult due to the nonlinearity of unit normals. We draw inspiration from previous works and apply a change of variables to accelerate the computation: instead of di-

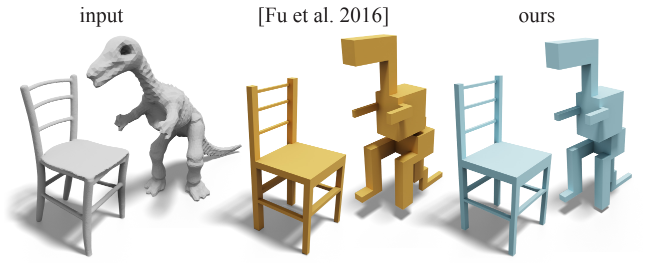


Figure 2: *Our method can be used to create PolyCube shapes (blue) and obtain comparable results to [FBL16] (yellow).*

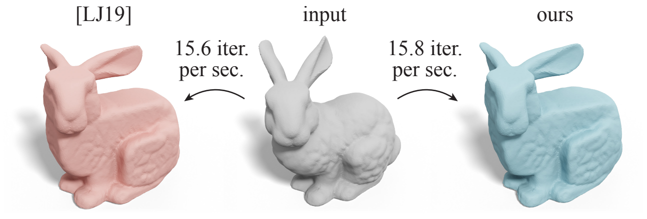


Figure 3: *Although being more general for creating different geometric styles (e.g., Fig. 1), our normal driven editing can also be applied to cubic stylization [LJ19], achieving comparable performance (blue) to the previous method (red).*

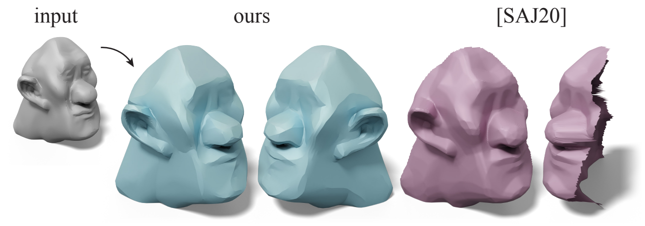


Figure 4: *Sellán et al. [SAJ20] propose a technique to make 2D heightfields developable (purple). In contrast, our method can create developable approximations for surface meshes in 3D (blue).*

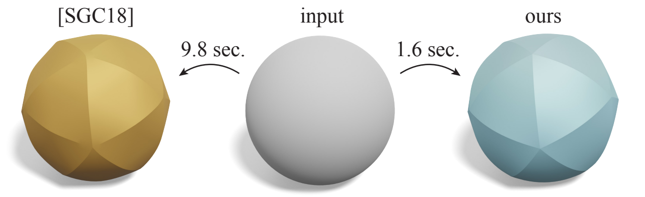


Figure 5: *Compared to the method proposed by Stein et al. [SGC18] for creating developable approximations (left), our method can create visually comparable results (right) with significant speed-ups.*

rectly optimizing the vertex positions, we optimize a set of rotations that rotate the normals of the input mesh to the set of desired normals. Our simple formulation with the change of variables results in a generic stylization algorithm that runs at interactive rates.

2. Related Work

Our work shares similar motivations to computer-assisted image stylization pioneered by Haeberli [Hae90]. But since our outputs are stylized 3D geometries, we focus the discussion on geometric stylization and geometric deformation methods.

Analogy-based Geometric Stylization

Many generative models have been proposed for creating stylized 3D objects, such as collage art [GSP*07; TRAS07], manga style [SLHC12], cubic style [LJ19], and neuronal homunculus [RRS12]. However, these methods are tailor-made for only a specific style.

Analogy $A : A' :: B : B'$ is a powerful idea to achieve different stylization results within a single framework. This idea has inspired several design tools for images [HJO*01], non-photorealistic renderings [SMGG01; FJL*16], and curves [HOCS02]. Beyond 2D data, the idea of analogy has also been used for transferring 3D geometric details from one shape to another. We omit the discussion on methods that are not based on analogies, such as mesh cloning [ZHW*06; TSS*11] and geometric learning [LKC*20; HHGC20; WAK*20; CKF*21; LZ21], and focus on analogy-based techniques. Ma et al. [MHS*14] propose a method for 3D style transfer based on patch-based assembly. However, their method cannot handle free-form deformations and requires the source and the exemplar shape to share a similar structure in order to compute high-quality correspondences. Bhat et al. [BIT04] propose a voxel-based texture synthesis method for transferring geometric details encoded in the volumetric grid. Berkiten et al. [BHS*17] use metric learning for details represented as displacement maps. These methods are designed for high-frequency details (e.g., wrinkles on the surface). In contrast, our spherical shape analogies focuses on larger scale free-form deformations. Albeit limited — in our analogies A is restricted to the unit sphere — our method enables a first step in this exciting direction.

Surface Normals in Shape Deformation

A key insight of our spherical shape analogies is to leverage surface normals to capture geometric styles. The surface normal is a fundamental geometric quantity and is ubiquitous in geometry processing. A representative example is in the *PolyCube* deformation [THCM04] where the goal is to optimize surface normals to be axis-aligned. Gregson et al. [GSZ11] and Zhao et al. [ZLL*17] use the closest rotation from the surface normal to an axis-aligned direction to drive the PolyCube deformation. Huang et al. [HJS*14] and Fu et al. [FBL16] propose to minimize energies defined on normals to create PolyCube shapes. In architectural geometry design, surface normals are a main ingredient to characterize polygon meshes with planar faces. The methods proposed by Deng et al. [DPW11] and Poranne et al. [POG13] utilize normals to formulate a *distance-from-plane* constraint to encourage planarity. Tang et al. [TSG*14] use the dot product between a face normal and its adjacent edge vectors to determine whether the vertices of a polygon are coplanar. Characterizing whether a mesh can be flattened to 2D without stretching or shearing, a.k.a. *developability*, also relies on surface normals. Stein et al. [SGC18] characterize discrete developability based on the 1-ring face normals, and propose an algorithm to compute piecewise developable surfaces. Sellán et al. [SAJ20] reformulate the developable energy into a convex semidefinite program for finding piecewise developable heightfields. In addition to these examples, deforming shapes into the cubic style [LJ19; FMR20], constructing shape abstractions [Ale21], surface parameterization [ZSL*20], and interactive mesh editing [YZX*04; SCL*04] are all related to surface normals. Many more examples can be found in the design of geometric filters, such as the Guided filter [ZDZ*15], the Shock filter [PK15], the Bilateral normal filter [ZFAT11], and the Total Variation mesh denoising [ZWZD15].

Our method can be adapted to these normal-based deformations. Compared to the PolyCube method [FBL16], we achieve comparable quality (see Fig. 2), but we can further generalize to polytopes (see Fig. 18). Compared to [LJ19] in cubic stylization (see Fig. 3), we can achieve similar performance, but we can further generalize to many styles other than the cubic style (see Fig. 1). In developable surface approximation, in contrast to the method by Sellán et al. [SAJ20], our method can be applied to surface triangle meshes (see Fig. 4) and is significantly faster than the method by Stein et al. [SGC18] (see Fig. 5).

Shape Deformation

Our geometric stylization method can also be perceived as a type of shape deformation method. We share technical similarities with methods that deform a shape while addressing given modeling constraints. A common choice is to minimize the *as-rigid-as-possible* (ARAP) energy [SA07; IMH05; CPSS10] while satisfying the constraints. This ARAP energy measures the rigidity of local surface patches and favors detail-preserving smooth deformations. In the case where locally rigid deformations are too constrained, the conformal energy [CPS11; VMW15] which preserves angles is commonly used. In contrast to ARAP, the conformal energy often triggers larger deformations as it allows both local uniform scaling and rigid transformations. In addition to mesh deformations, similar energies have also been used for parameterization [LZX*08], shape optimization [BDS*12], and simulating mass-spring systems [LBOK13]. The ARAP and conformal energies are also commonly used as regularization terms in mesh optimization problems, such as reconstruction [ZNI*14], surface registration [HAWG08; YMYK14], PolyCube construction [HJS*14], and surface stylization [LJ19]. Their popularity comes from the property that they favor smooth deformations and are amenable to fast optimizations. For the same reasons, we also use these as our regularization energies for interactive modeling tasks (see Fig. 11).

3. Spherical Shape Analogies

Our main idea is to use surface normals to capture the style of 3D objects: if two shapes share a similar normal "profile", we consider them to exhibit the same geometric style. Centered around this observation, as discussed in Sec. 1, we propose an analogy-based stylization method to translate the relationship between the normals of A', A to create a stylized output shape B' (see Fig. 6). Throughout the paper, we use green color to denote the target style shape A', gray color to denote the input shape B, and blue color to denote the output stylized shape B'.

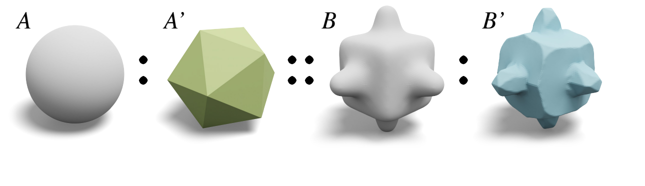


Figure 6: *We generate an output shape B' that relates to the input B in the same way as how the surface normal of a given primitive A' relates to the surface normal of a sphere A.*

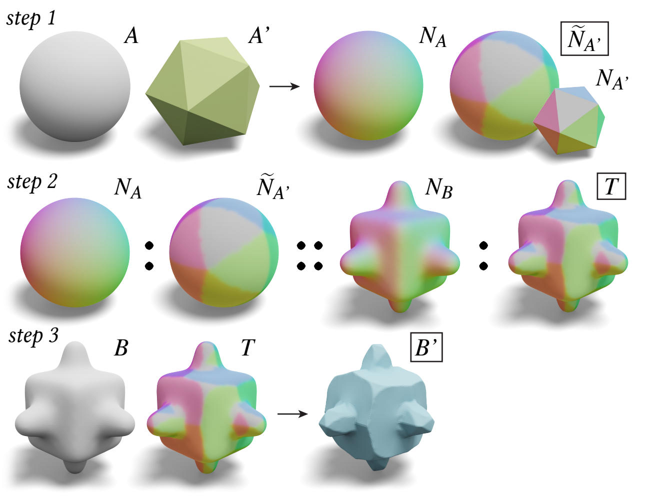


Figure 7: *Our algorithm defines the analogous relation based on the surface normals. We first map the normals of the style shape $N_{A'}$ to a unit sphere to obtain $\widetilde{N}_{A'}$ (top row), transfer the relationship between N_A and $\widetilde{N}_{A'}$ to the input shape to obtain the target normals T (middle row), then optimize the input shape B so that the actual output normals are aligned with the target normal T (bottom row).*

Our algorithm consists of three simple steps, described in Fig. 7: (1) we map the surface normal of A' to a unit sphere A in order to compute target normals on a sphere $\widetilde{N}_{A'}$, (2) we construct analogous target normals T that relate to N_B the same way $\widetilde{N}_{A'}$ relate to N_A, (3) we take B, T as inputs and generate the stylized shape B' whose normals approximate T via optimization.

3.1. Generating $\widetilde{N}_{A'}$

Depending on the provided style shape A' or user preferences, we consider three ways to get a set of target normals on a sphere $\widetilde{N}_{A'}$.

1. Closest normals. The simplest case is when the style shape A' is a simple convex shape with only few distinct face normals (e.g., icosahedron). We compute $\widetilde{N}_{A'}$ simply via snapping the normals of the sphere N_A to the nearest normal in the style shape $N_{A'}$.

2. Spherical parameterization. For a generic genus-0 shape (e.g., smooth or concave), we compute its parameterization to a sphere using, for example, conformalized mean curvature flow [KSB12]. Then $\widetilde{N}_{A'}$ can be computed from the spherical parameterization.

3. Normal Capture. If one desires more control, one can manually specify $\widetilde{N}_{A'}$, in the spirit of how *MatCap* (material capture

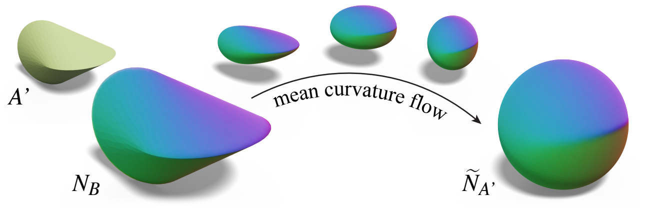
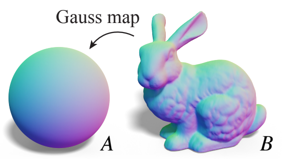


Figure 8: *Given a style shape A', we run mean curvature flow [KSB12] to map the normals of style shape $N_{A'}$ to a sphere as $\widetilde{N}_{A'}$.*

[SMGG01]) is used in rendering. We can then skip the first step in Fig. 7 and move on to the second step using the user-provided $\widetilde{N}_{A'}$.

3.2. Generating T



Generating target normals T on the input shape B using analogy requires the correspondences between A, B. We compute the map using the *Gauss map*, leveraging the fact that our A is always a unit sphere (see the inset, where we use colors to visualize the correspondences). Specifically, the unit normal vector of each element (e.g., vertex or face) on the input shape B can be equivalently interpreted as a point on the unit sphere A. Thus, we can easily map signals from A back to B. Once the correspondences are obtained via the normals of input shape N_B, we can trivially compute T by "pasting" $\widetilde{N}_{A'}$ on top of B.

3.3. Generating B'

After obtaining a set of target normals $T = \{\mathsf{t}_k\}$ for each vertex k, our goal is to obtain a deformed output shape B' whose surface normals approximate T. Let V be a matrix of vertex locations with size $|\mathsf{V}|$-by-3 and F be the face list with size $|\mathsf{F}|$-by-3 of the input shape B. Our output shape B' is a deformed version of the input shape and we use V' to denote the $|\mathsf{V}|$-by-3 matrix of the deformed vertex locations. We formulate the normal-driven deformation as an energy optimization in the following form:

$$\min_{\mathsf{V}'} \sum_{k \in \mathsf{V}} E_R(\mathsf{v}_k, \mathsf{u}_k) + \lambda a_k \|\hat{n}_k(\mathsf{V}') - \mathsf{t}_k\|_2^2, \qquad (1)$$

where E_R denotes a regularization energy to preserve the details of the input mesh, and the second part measures the squared distance from the output unit surface normal $\hat{\mathsf{n}}_k(\mathsf{V}')$ to the target output normal t_k at vertex k. We use a_k to denote the Voronoi area of the vertex k, λ is a weighting parameter to control the balance between the two terms, and v_k (u_k) is the input (output) location of vertex k. In Fig. 9, we can observe that using a small λ, the method preserves the input shape B. Using a bigger λ, the method favors in deforming the shape more into the style of A'. The choice of E_R depends on the user's intent. One can apply different regularizations to obtain different results. For the purposes of this exposition, we introduce our optimization based on ARAP regularization in Sec. 3.4. We discuss how to extend to other regularizations in Sec. 4.1.

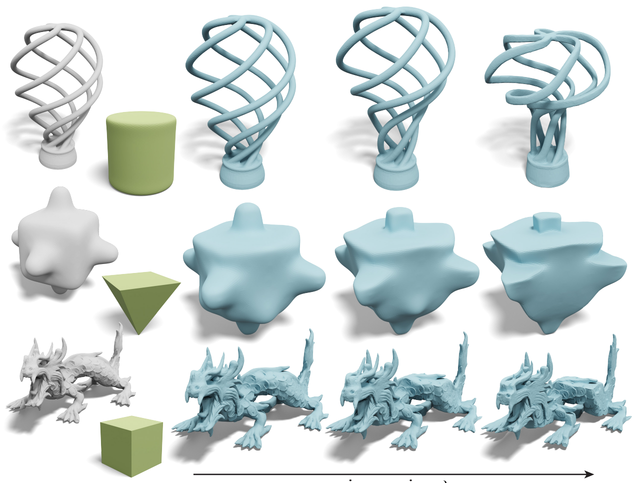
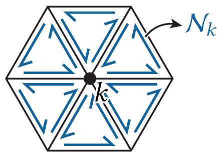


Figure 9: *The λ parameter in Eq. 3 controls the balance between preserving the original shape and satisfying the desired style. We show different stylization results with increasing λ. ©Spiral Light Bulb (top) by benglish under CC BY-SA.*

3.4. Normal-Driven Optimization with ARAP

We use $\mathsf{e}_{ij} := \mathsf{v}_j - \mathsf{v}_i \in \mathbb{R}^3$ to denote the edge vector between vertices i, j on the original mesh, and $\mathsf{e}'_{ij} := \mathsf{v}'_j - \mathsf{v}'_i$ for the edge vectors on the deformed mesh. We can write down the energy that uses ARAP regularization as

$$\min_{\mathsf{V}',\mathsf{R}} \sum_{k \in V} \underbrace{\sum_{i,j \in \mathcal{N}_k} w_{ij} \|\mathsf{R}_k \mathsf{e}_{ij} - \mathsf{e}'_{ij}\|_2^2}_{E_{\text{ARAP}}} + \lambda a_k \|\hat{n}_k(\mathsf{V}') - \mathsf{t}_k\|_2^2, \qquad (2)$$



We use $\mathcal{N}_k$ to denote the edge vectors of the *spokes and rims* at vertex k (see the inset) [CPSS10], $\mathsf{R}_k \in SO(3)$ to denote a 3-by-3 rotation matrix defined on k, and w_{ij} is the cotangent weight of edge i, j [PP93]. However, this energy is difficult to optimize because the term $\hat{n}_k(\mathsf{V}')$ is non-linear in V'.

We adapt the observation made in [LJ19] that the space of unit vectors can be captured by rotations. Thus, we can perform a change of variables by replacing $\hat{n}_k(\mathsf{V}')$ with the *rotated* unit normal of the input mesh $\mathsf{R}_k \hat{\mathsf{n}}_k$ as

$$\boxed{\min_{\mathsf{V}',\mathsf{R}} \sum_{k \in V} \sum_{i,j \in \mathcal{N}_k} w_{ij} \|\mathsf{R}_k \mathsf{e}_{ij} - \mathsf{e}'_{ij}\|_2^2 + \lambda a_k \|\mathsf{R}_k \hat{\mathsf{n}}_k - \mathsf{t}_k\|_2^2,} \qquad (3)$$

where $\hat{\mathsf{n}}_k$ is the kth unit vertex normal of the input mesh computed via area-weighted average of face normals, which is constant throughout the optimization. This $\mathsf{R}_k \hat{\mathsf{n}}_k$ can be perceived as an approximation of the area-weighted vertex normals of the output mesh $\hat{n}_k(\mathsf{V}')$. In Fig. 10, we visualize the difference between the output normals $\hat{n}_k(\mathsf{V}')$ and the rotated input normals $\mathsf{R}_k \hat{\mathsf{n}}_k$. We can notice that $\mathsf{R}_k \hat{\mathsf{n}}_k$ is a decent approximation of the output vertex normals computed via area-weighted average. We can observe that error tend to concentrate on high-curvature regions because

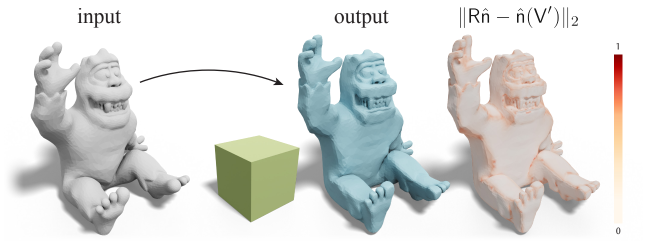

input
output
$\|\mathsf{R}\hat{\mathsf{n}} - \hat{\mathsf{n}}(\mathsf{V}')\|_2$
1
0

Figure 10: *Empirically, we show that rotated input normals $\mathsf{R}_k\hat{\mathsf{n}}_k$ is a good approximation of the area-weighted output vertex normals $\hat{\mathsf{n}}_k(\mathsf{V}')$. We can observe that the error mostly occurs on the high-curvature regions (right). ©Proto Paradigm under CC BY.*

discrete vertex normals are less accurate along on those regions and the ARAP regularization encourages smooth deformation. This change of variables allows us to solve for R_ks in parallel and make this energy quadratic in V'. In addition, the fact that R_k is shared across the ARAP term and the normal term enables us to jointly consider both the regularization and the normal terms when obtaining the deformed vertex locations V'.

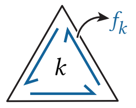
We minimize this energy via the local/global strategy [SA07], where the local step involves solving a set of small *Orthogonal Procrustes* problems and the global step amounts to a linear solve. For the sake of reproducibility, we reiterate the local-global steps for our energy in App. A, B. Non-linear methods, such as Newton's method, could be applied to our scenario. It is however far slower than the local-global optimization since a single iteration of the Newton's method could be more expensive than 100 iterations of the local-global iterations (see [LBK17]). Thus, it is less suitable for our interactive applications. Further accelerating our solver using other optimization methods (e.g., [KGL16; PDZ*18; ZBK18]) should be possible, but is left as future work.

4. Extensions & Analysis

In this section, we introduce its extensions to different regularizations and how to handle cases where target normals $T(B')$ are a function of output geometry.

4.1. Different Regularizations

In addition to E_{ARAP}, the normal-driven optimization supports different regularization energies for different modeling intents. One could use ARAP when the goal is to produce a smooth deformation that preserves surface details. If one wants to produce a non-smooth deformation (e.g., sharp creases) while preserving local rigidity, one could instead use a *face-only* ARAP energy E_{FARAP} discussed in [ZG16; LG15] which consists of only the membrane term. If one is interested in preserving the textures and allowing local scaling, one could use an *as-conformal-as-possible* energy E_{ACAP} [BDS*12].



Face-only ARAP. The core idea is to remove the bending term from ARAP and only measure the membrane term [TPBF87], so that two adjacent triangles can bend freely. We achieve this by applying the idea from [ZG16; LG15] which only measures the ARAP energy over the three edge vectors of a face f_k (see the inset), instead of the spokes and rims $\mathcal{N}_k$. Precisely, we can write this "face-only" ARAP regularization E_{FARAP} as

$$E_{\text{FARAP}}(\mathsf{V}', \mathsf{R}) = \sum_{k \in F} \sum_{i,j \in f_k} w_{ij} \|\mathsf{R}_k \mathsf{e}_{ij} - \mathsf{e}'_{ij}\|_2^2, \tag{4}$$

As-conformal-as-possible. If the goal is to create novel geometric details, it is crucial to allow non-rigid deformations. However, an arbitrary deformation may lead to undesirable behaviors, such as badly shaped triangles. Thus constraining the angle preservation, a.k.a. conformality, will be a suitable regularization. Specifically, we use the ACAP energy E_{ACAP} in [BDS*12] as our regularization

$$E_{\text{ACAP}}(\mathsf{V}', \mathsf{R}, \mathsf{s}) = \sum_{k \in F} \sum_{i,j \in \mathcal{N}_k} w_{ij} \|s_k \mathsf{R}_k \mathsf{e}_{ij} - \mathsf{e}'_{ij}\|_2^2, \tag{5}$$

where s_k is a scalar representing the scaling of local patch. One can compute the optimal s_k analytically via the method by Schönemann et al. [SC70] (see App. C).

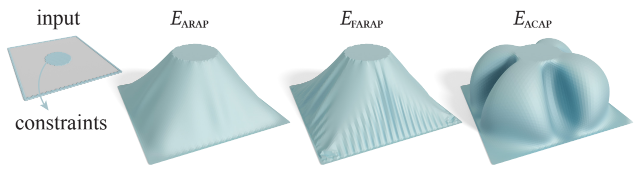


Figure 11: *Different regularizations favor different behaviors. Given a sheet (gray), we pull up the center part (central blue dots) and shrink the boundary (blue dots on the boundary), then we minimize each regularization energy to determine the unconstrained vertices. We can observe that E_{ARAP} favors rigid and smooth interpolation, E_{FARAP} favors sharp bending between triangles, and E_{ACAP} favors to preserve angles while allowing local scaling.*

Deploying these regularizations $E_{\text{FARAP}}, E_{\text{ACAP}}$ requires only a few changes in the optimization steps. Deploying E_{FARAP} only involves changing the incidence matrix. Deploying E_{ACAP} only requires adding one more line of code in the local step to solve an *isotropic orthogonal Procrustes* problem [SC70]. We detail such changes in App. C. In Fig. 11, we apply the same deformation to a sheet but with different regularizations. We can perceive that different regularizations favor drastically different solutions.

Our framework allows one to easily plug-and-play different regularizations. Specifically, we use E_{ARAP} for applications that favor smooth deformation (e.g., Fig. 13), E_{FARAP} for creating sharp creases (Fig. 18, 20), and E_{ACAP} when one wants to manipulate geometric details such as in Fig. 22.

4.2. Dynamic Target Normals

Our method converges to a local minimum. Empirically, we observe that treating the target normal T as a constant throughout the optimization may work fine perceptually in many cases (see the left pair in Fig. 12). However, constant T may lead to an undesirable local minimum due to a sub-optimal assignment of T (see the right pair in Fig. 12). Inspired by Projective Dynamics [BML*14], a simple solution to avoid such local minima is to treat T as a function of B' ($N_{B'}$ specifically), and update T at every iteration. We

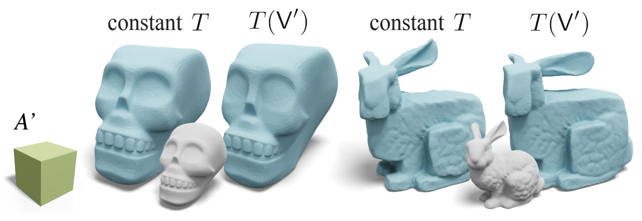
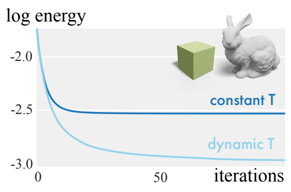


Figure 12: *Setting the target normal T as a constant or treating it as a function of the output mesh $T(B')$ leads to different local minima. In many cases (left pair), both options lead to similar looking results. But setting T as a constant may result in an undesirable local minimum in some cases (right), such as the ears of the bunny.*

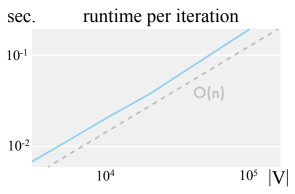
Algorithm 1: Normal-driven optimization

```
Input  : A triangle mesh $\mathsf{V}, \mathsf{F}$ and a weight $\lambda$
Output: Deformed vertex positions $\mathsf{V}'$
1. compute $\widetilde{N}_{A'}$                          // step 1, Sec. 3.1
2. compute $T$ from $\hat{n}(\mathsf{V})$                // step 2, Sec. 3.2
3. $\hat{\mathsf{n}} \leftarrow \hat{n}(\mathsf{V})$     // compute input surface normals
4. $\mathsf{Q}, \mathsf{K} \leftarrow precompute(\mathsf{V}, \mathsf{F})$   // see App. B
5. $\mathsf{V}' \leftarrow \mathsf{V}$
6. while not converge do
7.     $\mathsf{R} \leftarrow local\text{-}step(\mathsf{V}', \hat{\mathsf{n}}, T, \lambda)$   // App. A
8.     $\mathsf{V}' \leftarrow global\text{-}step(\mathsf{R}, \mathsf{Q}, \mathsf{K})$   // App. B
9.     compute $T$ from $\hat{n}(\mathsf{V}')$           // (optional) for dynamic $T$
```

summarize the pseudo code in Alg. 1. If T is a constant throughout the optimization, one can simply skip the optional step at line 9.



In terms of convergence, in the case where T is constant, the convergence behaves the same as the original ARAP [SA07], where the energy decreases monotonically. In the case where T is dependent to B', we do not guarantee a monotonic decrease in energy, but the optimization still converges in our experiments. In the inset, we visualize the convergence plot for examples in Fig. 12.



We implement our algorithm in C++ with Eigen [GJ*10] and evaluate our method on a MacBook Pro with an Intel i5 2.3GHz processor. Our method runs 24 iterations per second for a mesh with around 20k vertices. We report a complete picture of our runtime in the inset. The local step will be the computation bottleneck for meshes with less than 20k vertices, but further acceleration can be achieved via the method by Zhang et al. [ZJA21]. Typically, within the first 10 iterations, our method can achieve a visually similar result compared to the converged solution. This property enables us to build an interactive tool for users to play with different style shapes A' or artistic controls.

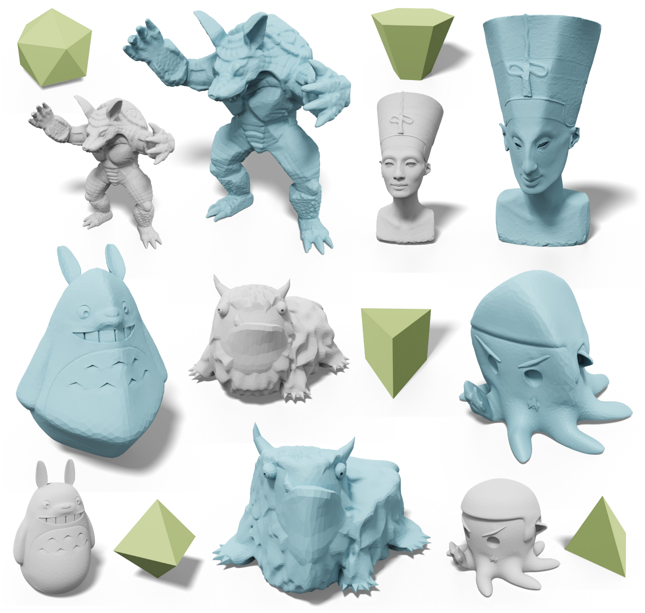


Figure 13: *Given an input shape (gray), our approach can transfer the style of a primitive shape (green) to obtain a stylized output shape (blue). ©Johannes (bottom left), Joseph Larson (bottom middle), and Angelo Tartanian (bottom left) under CC BY. The Nefertiti mesh (top right) was scanned by Nora Al-Badri and Jan Nikolai Nelles from the Nefertiti bust.*

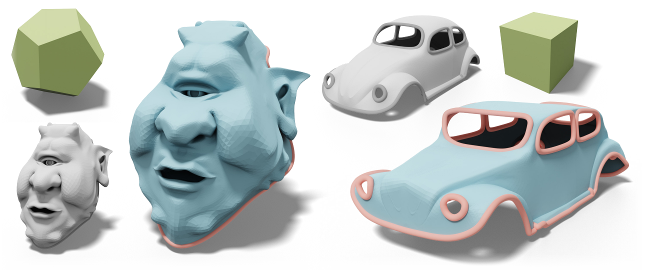


Figure 14: *Even for input shapes with boundaries (gray), our method is still applicable to transfer the style of primitive shapes (green) to obtain the stylized output shape (blue).*

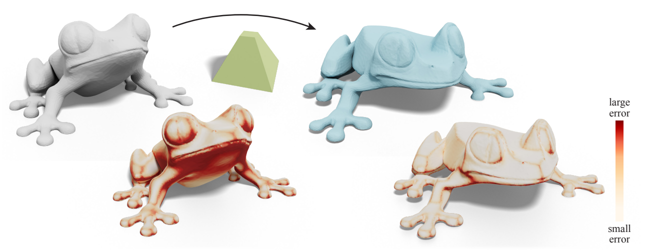


Figure 15: *We visualize the difference between the mesh normals and the normals of the style shape. Our normal-driven optimization effectively reduce the difference to the target normals. ©Morena Protti under CC BY.*

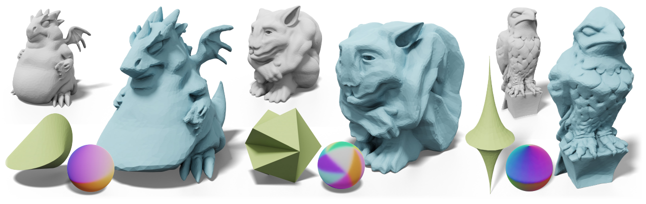


Figure 16: *When the input shape is smooth or non-convex, we use the mean curvature flow (see Fig. 8) to obtain target normals to proceed the optimization. We deform the input shapes (gray) to exhibit the style of an oloid (left, green), a Jessen's icosahedron (middle, green), and a tractricoid (right, green), respectively. ©Splotchy Ink (left), fong182 (middle), and Colin Freeman (right) under CC BY.*

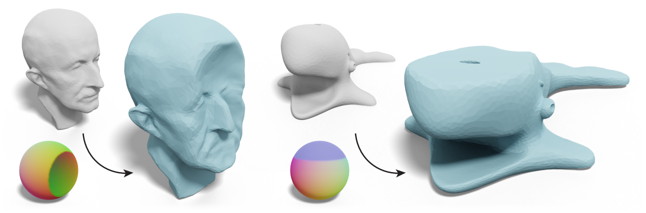


Figure 17: *One can manually specify the target normals on a sphere (Normal Captures) for full control, and deform the input shape (gray) to the style (blue) prescribed by the colored sphere. ©MakerBot (right) under CC BY.*

5. Applications

The major benefit of our analogy-based stylization method is that one can plug-and-play different style shapes to obtain different results. When one provides convex primitives with few distinct face normals, we can simply use the method discussed in Sec. 3.1 to turn an input shape into the style of the primitive (see Fig. 13, 14). In Fig. 15, we also quantitatively show that our method can effectively reduce the difference between mesh normals and the normals of a primitive. If the provided style shape is smooth or non-convex, where the simple closest normal may fail to capture the style, one could use a spherical parameterization described in Sec. 3.1 to achieve the stylization. If desiring more user controls, one could "paint" the desired surface normals on a unit sphere (see Sec. 3.1), and then transfer the style of the painted normals directly to the input (see Fig. 17)

5.1. PolyCube Deformation

If one is interested in PolyCube maps [THCM04], we can adapt normal driven editing to create PolyCube maps,following the observation in [ZLL*17]. Specifically, we need to use a cube as a style shape and move the pre-computation step in Alg. 1 to the optimization loop. Moving the pre-computation in the loop would no longer preserve the original details, which is desirable for creating PolyCube shapes. This modification may also lead to badly shaped triangle. When these faces appear, a quick solution is to move the vertex towards the 1-ring average by a small amount to improve triangle quality. For the sake of comparison, we use the same PolyCube segmentation as in [FBL16] and show that we can

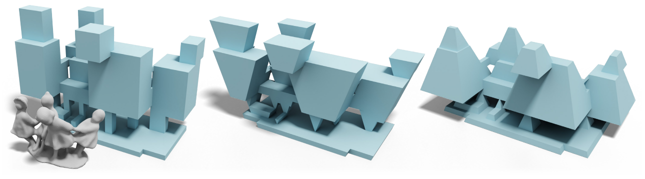


Figure 18: *By using different sets of normals, we can generalize the PolyCube method (left) to create polygonal boxed maps.*

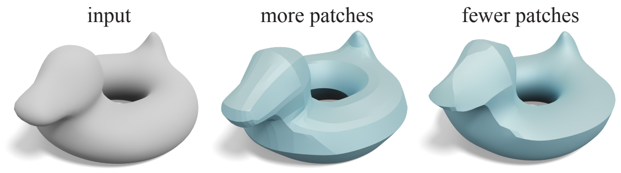


Figure 19: *Stein et al. [SGC18] control the patches on the developable surfaces via remeshing the input. We, instead, can control the amount of creases (middle, right) by tuning a single parameter (see App. E).*

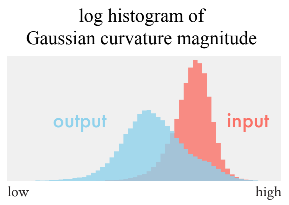
achieve comparable results in Fig. 2. We can further generalize the PolyCube map to other polygonal boxes by specifying non-cube normals (see Fig. 18).

5.2. Developable Surface Approximation

So far we have only considered an explicit shape or a set of painted normals as our style shape. Here we further extend our method to support an energy that describes a certain style. In particular, we consider the target normal T is computed via an optimization

$$T = \arg\min_{T} f(B'), \tag{6}$$

and, similar to the case where T is dependent to B', we update T at every iteration in the local/global solve.



We evaluate this extension via setting f to be the discrete developability energy proposed in [SGC18], with details provided in App. E. Compared to the original method, our approach contains a regularization term in addition to the developable energy, thus our optimization requires no remeshing and results in the faster optimization (see Fig. 5). In Fig. 19, we further show that our framework enables one to control the number of creases in the results. With our framework one can interactively create a variety of piece-wise developable shapes (see Fig. 20). In Fig. 21, we evaluate our results by visualizing the discrete Gaussian curvature before and after running our developable flow. We can observe that the Gaussian curvature concentrates along the creases and results in a piece-wise developable surface. In the inset, we quantitatively demonstrate that our method effectively increases the developability of the mesh in Fig. 21.

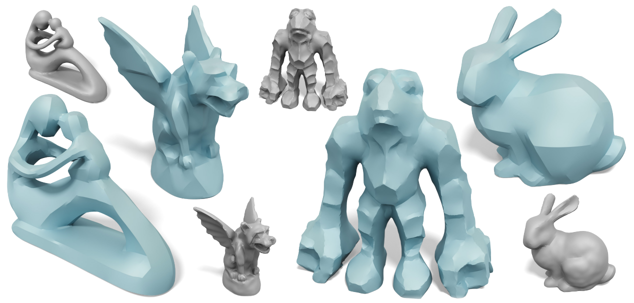


Figure 20: *Our normal driven editing can be used to create many piece-wise developable surfaces (blue). Our method requires no remeshing and is fast enough for interactive modeling. ©cerberus333 (third) under CC BY-NC.*

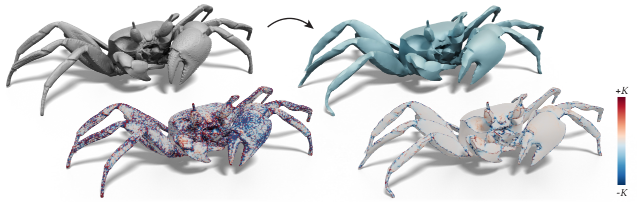
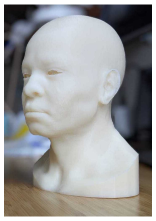


Figure 21: *We use our normal driven editing to deform the input shape (gray) into a piece-wise developable approximation (blue). In the bottom row, we visualize the Gaussian curvature concentrates on the creases after the deformation, leading to a piece-wise developable shape. ©Oliver Laric under CC BY-NC-SA.*

6. Limitations & Future Work

Our method draws inspiration from Projective Dynamics [BML*14] to handle the case where target normals T are a function of output shape B' (e.g., Fig. 12, 20). Although being fast and suitable for our intended interactive applications, it often struggles to converge to a highly accurate solution. Extending our optimization to, for example, Newton's method would be desirable for applications that desire highly accurate solutions.

Our approach is restricted to a sphere as our reference shape A, and uses the Gauss map to determine the correspondences between A and the input B. As the Gauss map purely relies on surface normals to determine the map, the resulting map is ignorant to area distortion. This characteristic is beneficial to handle input shapes B that are very different (e.g., different genus) from a sphere because in these cases it is challenging to obtain a map with low area distortion. However, the price we have to pay is that we cannot support structured and high-frequency patterns (e.g., geometric texture synthesis). Thus, if one is interested in stylizing shapes with detailed textures, we suggest to first synthesize target normals on the surface directly [WLKT09] then perform the normal-driven optimization (Sec. 3.4). In Fig. 22 we demonstrate this alternative by unbaking an existing normal map for manufacturing purposes (see the inset) and synthesizing normal textures from an image.



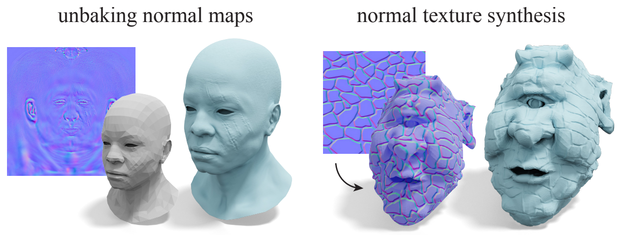


Figure 22: *If one is interested in creating high-frequency geometric textures, we recommend to compute target normals via texture synthesis and then optimize the geometry via the normal-driven optimization. We demonstrate an example of "unbaking" normal maps (left) and an example of geometric texture synthesis (right).*

Our method currently supports manifold triangle meshes. Extending to non-manifold meshes, polygon meshes, volumetric meshes, and point clouds could be beneficial to handle real-world geometric data. Not every shape or normal capture sphere is valid to serve as the style shape of our algorithm. Discovering the validity of a style shape is important to understand the behavior of these novel modeling methods. Removing the assumption about the source shape being a sphere could lead to a more general analogy-based shape editing. Based on the observation that surface normals are a promising geometric quantity to capture the style of a shape. Developing a better categorization of styles based on normals or exploring learning-based techniques on normals (instead of vertices) could lead to novel stylization methods.

Acknowledgements

Our research is funded in part by NSERC Discovery (RGPIN2017–05235, RGPAS–2017–507938), New Frontiers of Research Fund (NFRFE–201), the Ontario Early Research Award program, the Canada Research Chairs Program, the Fields Centre for Quantitative Analysis and Modelling and gifts by Adobe Systems, Autodesk and MESH Inc. We thank Sheldon Andrews, Abhishek Madan, Silvia Sellán, Oded Stein, Li-Yi Wei for helps on experiments. We thank members of Dynamic Graphics Project at the University of Toronto; Sarah Kushner, Abhishek Madan, Silvia Sellán, Letícia Mattos Da Silva, Towaki Takikawa for proofreading; John Hancock for the IT support. We thank all the artists for sharing a rich variety of 3D models.

References

[Ado21] Adobe Inc. *Adobe Photoshop*. Version 22.3.1. 2021. url: `https://www.adobe.com` 1.

[Ale21] Alexa, Marc. "PolyCover: Shape Approximating with Discrete Surface Orientation". *IEEE Computer Graphics and Applications* (2021) 3.

[BDS*12] Bouaziz, Sofien, Deuss, Mario, Schwartzburg, Yuliy, et al. "Shape-Up: Shaping Discrete Geometry with Projections". *Comput. Graph. Forum* 31.5 (2012), 1657–1667 3, 5.

[BHS*17] BERKITEN, SEMA, HALBER, MACIEJ, SOLOMON, JUSTIN, et al. "Learning Detail Transfer based on Geometric Features". *Comput. Graph. Forum* 36.2 (2017), 361–373 2.

[BIT04] BHAT, PRAVIN, INGRAM, STEPHEN, and TURK, GREG. "Geometric Texture Synthesis by Example". *Second Eurographics Symposium on Geometry Processing, Nice, France, July 8-10, 2004*. Ed. by BOISSONNAT, JEAN-DANIEL and ALLIEZ, PIERRE. Vol. 71. ACM International Conference Proceeding Series. Eurographics Association, 2004, 41–44 2.

[BML*14] BOUAZIZ, SOFIEN, MARTIN, SEBASTIAN, LIU, TIANTIAN, et al. "Projective dynamics: fusing constraint projections for fast simulation". *ACM Trans. Graph.* 33.4 (2014), 154:1–154:11 5, 8, 11.

[CKF*21] CHEN, ZHIQIN, KIM, VLADIMIR G., FISHER, MATTHEW, et al. "DECOR-GAN: 3D Shape Detailization by Conditional Refinement". *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2021) 2.

[CPS11] CRANE, KEENAN, PINKALL, ULRICH, and SCHRÖDER, PETER. "Spin transformations of discrete surfaces". *ACM Trans. Graph.* 30.4 (2011), 104 3.

[CPSS10] CHAO, ISAAC, PINKALL, ULRICH, SANAN, PATRICK, and SCHRÖDER, PETER. "A simple geometric model for elastic deformations". *ACM Trans. Graph.* 29.4 (2010), 38:1–38:6 3, 4.

[DPW11] DENG, BAILIN, POTTMANN, HELMUT, and WALLNER, JOHANNES. "Functional webs for freeform architecture". *Comput. Graph. Forum* 30.5 (2011), 1369–1378 2.

[FBL16] FU, XIAO-MING, BAI, CHONG-YANG, and LIU, YANG. "Efficient Volumetric PolyCube-Map Construction". *Comput. Graph. Forum* 35.7 (2016), 97–106 2, 3, 7.

[FJL*16] FISER, JAKUB, JAMRISKA, ONDREJ, LUKÁC, MICHAL, et al. "StyLit: illumination-guided example-based stylization of 3D renderings". *ACM Trans. Graph.* 35.4 (2016), 92:1–92:11 2.

[FMR20] FUMERO, MARCO, MÖLLER, MICHAEL, and RODOLÀ, EMANUELE. "Nonlinear spectral geometry processing via the TV transform". *ACM Trans. Graph.* 39.6 (2020), 199:1–199:16 3.

[GJ*10] GUENNEBAUD, GAËL, JACOB, BENOÎT, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010 6.

[GSP*07] GAL, RAN, SORKINE, OLGA, POPA, TIBERIU, et al. "3D collage: expressive non-realistic modeling". *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*. ACM. 2007, 7–14 2.

[GSZ11] GREGSON, JAMES, SHEFFER, ALLA, and ZHANG, EUGENE. "All-Hex Mesh Generation via Volumetric PolyCube Deformation". *Comput. Graph. Forum* 30.5 (2011), 1407–1416 2.

[Hae90] HAEBERLI, PAUL. "Paint by numbers: abstract image representations". *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1990, Dallas, TX, USA, August 6-10, 1990*. Ed. by BASKETT, FOREST. ACM, 1990, 207–214 2.

[HAWG08] HUANG, QI-XING, ADAMS, BART, WICKE, MARTIN, and GUIBAS, LEONIDAS J. "Non-Rigid Registration Under Isometric Deformations". *Comput. Graph. Forum* 27.5 (2008), 1449–1457 3.

[HHGC20] HERTZ, AMIR, HANOCKA, RANA, GIRYES, RAJA, and COHEN-OR, DANIEL. "Deep geometric texture synthesis". *ACM Trans. Graph.* 39.4 (2020), 108 2.

[HJO*01] HERTZMANN, AARON, JACOBS, CHARLES E., OLIVER, NURIA, et al. "Image analogies". *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001, Los Angeles, California, USA, August 12-17, 2001*. Ed. by POCOCK, LYNN. ACM, 2001, 327–340 1, 2.

[HJS*14] HUANG, JIN, JIANG, TENGFEI, SHI, ZEYUN, et al. "l1-Based Construction of Polycube Maps from Complex Shapes". *ACM Trans. Graph.* 33.3 (June 2014) 2, 3.

[HOCS02] HERTZMANN, AARON, OLIVER, NURIA, CURLESS, BRIAN, and SEITZ, STEVEN M. "Curve Analogies". *Proceedings of the 13th Eurographics Workshop on Rendering Techniques, Pisa, Italy, June 26-28, 2002*. Ed. by GIBSON, SIMON and DEBEVEC, PAUL E. Vol. 28. ACM International Conference Proceeding Series. Eurographics Association, 2002, 233–246 2.

[IMH05] IGARASHI, TAKEO, MOSCOVICH, TOMER, and HUGHES, JOHN F. "As-rigid-as-possible shape manipulation". *ACM Trans. Graph.* 24.3 (2005), 1134–1141 3.

[KGL16] KOVALSKY, SHAHAR Z., GALUN, MEIRAV, and LIPMAN, YARON. "Accelerated quadratic proxy for geometric optimization". *ACM Trans. Graph.* 35.4 (2016), 134:1–134:11 5.

[KSB12] KAZHDAN, MICHAEL, SOLOMON, JAKE, and BEN-CHEN, MIRELA. "Can Mean-Curvature Flow be Modified to be Non-singular?": *Comput. Graph. Forum* 31.5 (2012), 1745–1754 3, 4.

[LBK17] LIU, TIANTIAN, BOUAZIZ, SOFIEN, and KAVAN, LADISLAV. "Quasi-Newton Methods for Real-Time Simulation of Hyperelastic Materials". *ACM Trans. Graph.* 36.3 (2017), 23:1–23:16 5.

[LBOK13] LIU, TIANTIAN, BARGTEIL, ADAM W., O'BRIEN, JAMES F., and KAVAN, LADISLAV. "Fast simulation of mass-spring systems". *ACM Trans. Graph.* 32.6 (2013), 214:1–214:7 3.

[LG15] LEVI, ZOHAR and GOTSMAN, CRAIG. "Smooth Rotation Enhanced As-Rigid-As-Possible Mesh Animation". *IEEE Trans. Vis. Comput. Graph.* 21.2 (2015), 264–277 5.

[LJ19] LIU, HSUEH-TI DEREK and JACOBSON, ALEC. "Cubic stylization". *ACM Trans. Graph.* 38.6 (2019), 197:1–197:10 2–4.

[LKC*20] LIU, HSUEH-TI DEREK, KIM, VLADIMIR G., CHAUDHURI, SIDDHARTHA, et al. "Neural subdivision". *ACM Trans. Graph.* 39.4 (2020), 124 2.

[LZ21] LI, MANYI and ZHANG, HAO. "D^{2}IM-Net: Learning Detail Disentangled Implicit Fields from Single Images". *Proc. of CVPR*. 2021 2.

[LZX*08] LIU, LIGANG, ZHANG, LEI, XU, YIN, et al. "A Local/Global Approach to Mesh Parameterization". *Comput. Graph. Forum* 27.5 (2008), 1495–1504 3.

[MHS*14] MA, CHONGYANG, HUANG, HAIBIN, SHEFFER, ALLA, et al. "Analogy-driven 3D style transfer". *Comput. Graph. Forum* 33.2 (2014), 175–184 2.

[PDZ*18] PENG, YUE, DENG, BAILIN, ZHANG, JUYONG, et al. "Anderson acceleration for geometry optimization and physics simulation". *ACM Trans. Graph.* 37.4 (2018), 42:1–42:14 5.

[Pix20] PIXOLOGIC INC. *ZBrush*. Version 2021.5.1. 2020. URL: `https://pixologic.com` 1.

[PK15] PRADA, FABIAN and KAZHDAN, MISHA. "Unconditionally Stable Shock Filters for Image and Geometry Processing". *Comput. Graph. Forum* 34.5 (2015), 201–210 3.

[POG13] PORANNE, ROI, OVREIU, ELENA, and GOTSMAN, CRAIG. "Interactive Planarization and Optimization of 3D Meshes". *Comput. Graph. Forum* 32.1 (2013), 152–163 2.

[PP93] PINKALL, ULRICH and POLTHIER, KONRAD. "Computing Discrete Minimal Surfaces and Their Conjugates". *Experimental Mathematics* 2.1 (1993), 15–36 4, 11.

[RRS12] REINERT, BERNHARD, RITSCHEL, TOBIAS, and SEIDEL, HANS-PETER. "Homunculus Warping: Conveying importance using self-intersection-free non-homogeneous mesh deformation". *Comput. Graph. Forum* 31.7-2 (2012), 2165–2171 2.

[SA07] SORKINE, OLGA and ALEXA, MARC. "As-rigid-as-possible surface modeling". *Proceedings of the Fifth Eurographics Symposium on Geometry Processing, Barcelona, Spain, July 4-6, 2007*. Ed. by BELYAEV, ALEXANDER G. and GARLAND, MICHAEL. Vol. 257. ACM International Conference Proceeding Series. Eurographics Association, 2007, 109–116 3, 5, 6, 11.

[SAJ20] SELLÁN, SILVIA, AIGERMAN, NOAM, and JACOBSON, ALEC. "Developability of heightfields via rank minimization". *ACM Trans. Graph.* 39.4 (2020), 109 2, 3.

[SC70] SCHÖNEMANN, PETER H and CARROLL, ROBERT M. "Fitting one matrix to another under choice of a central dilation and a rigid motion". *Psychometrika* 35.2 (1970), 245–255 5, 11.

[SCL*04] SORKINE, OLGA, COHEN-OR, DANIEL, LIPMAN, YARON, et al. "Laplacian Surface Editing". *Second Eurographics Symposium on Geometry Processing, Nice, France, July 8-10, 2004*. Ed. by BOISSONNAT, JEAN-DANIEL and ALLIEZ, PIERRE. Vol. 71. ACM International Conference Proceeding Series. Eurographics Association, 2004, 175–184 3.

[SGC18] STEIN, ODED, GRINSPUN, EITAN, and CRANE, KEENAN. "Developability of triangle meshes". *ACM Trans. Graph.* 37.4 (2018), 77:1–77:14 2, 3, 7, 11.

[SLHC12] SHEN, LIANG-TSEN, LUO, SHENG-JIE, HUANG, CHUN-KAI, and CHEN, BING-YU. "SD Models: Super-Deformed Character Models". *Comput. Graph. Forum* 31.7-1 (2012), 2067–2075 2.

[SMGG01] SLOAN, PETER-PIKE J., MARTIN, WILLIAM, GOOCH, AMY, and GOOCH, BRUCE. "The Lit Sphere: A Model for Capturing NPR Shading from Art". *Proceedings of the Graphics Interface 2001 Conference, Ottawa, Ontario, Canada, June 7-9, 2001*. Canadian Human-Computer Communications Society, 2001, 143–150 1, 2, 4.

[THCM04] TARINI, MARCO, HORMANN, KAI, CIGNONI, PAOLO, and MONTANI, CLAUDIO. "PolyCube-Maps". *ACM Trans. Graph.* 23.3 (2004), 853–860 2, 7.

[TPBF87] TERZOPOULOS, DEMETRI, PLATT, JOHN C., BARR, ALAN H., and FLEISCHER, KURT W. "Elastically deformable models". *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1987, Anaheim, California, USA, July 27-31, 1987*. Ed. by STONE, MAUREEN C. ACM, 1987, 205–214 5.

[TRAS07] THEOBALT, CHRISTIAN, ROESSL, CHRISTIAN, AGUIAR, EDILSON DE, and SEIDEL, HANS-PETER. "Animation Collage". *Eurographics/SIGGRAPH Symposium on Computer Animation*. Ed. by METAXAS, DIMITRIS and POPOVIC, JOVAN. The Eurographics Association, 2007 2.

[TSG*14] TANG, CHENGCHENG, SUN, XIANG, GOMES, ALEXANDRA, et al. "Form-finding with polyhedral meshes made simple". *ACM Trans. Graph.* 33.4 (2014), 70:1–70:9 2.

[TSS*11] TAKAYAMA, KENSHI, SCHMIDT, RYAN M., SINGH, KARAN, et al. "GeoBrush: Interactive Mesh Geometry Cloning". *Comput. Graph. Forum* 30.2 (2011), 613–622 2.

[VMW15] VAXMAN, AMIR, MÜLLER, CHRISTIAN, and WEBER, OFIR. "Conformal mesh deformations with Möbius transformations". *ACM Trans. Graph.* 34.4 (2015), 55:1–55:11 3.

[WAK*20] WANG, YIFAN, AIGERMAN, NOAM, KIM, VLADIMIR G., et al. "Neural Cages for Detail-Preserving 3D Deformations". *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. IEEE, 2020, 72–80 2.

[WLKT09] WEI, LI-YI, LEFEBVRE, SYLVAIN, KWATRA, VIVEK, and TURK, GREG. "State of the Art in Example-based Texture Synthesis". *30th Annual Conference of the European Association for Computer Graphics, Eurographics 2009 - State of the Art Reports, Munich, Germany, March 30 - April 3, 2009*. Ed. by PAULY, MARC and GREINER, GÜNTHER. Eurographics Association, 2009, 93–117 8.

[YMYK14] YOSHIYASU, YUSUKE, MA, WAN-CHUN, YOSHIDA, EIICHI, and KANEHIRO, FUMIO. "As-Conformal-As-Possible Surface Registration". *Comput. Graph. Forum* 33.5 (2014), 257–267 3.

[YZX*04] YU, YIZHOU, ZHOU, KUN, XU, DONG, et al. "Mesh editing with poisson-based gradient field manipulation". *ACM Trans. Graph.* 23.3 (2004), 644–651 3.

[ZBK18] ZHU, YUFENG, BRIDSON, ROBERT, and KAUFMAN, DANNY M. "Blended cured quasi-newton for distortion optimization". *ACM Trans. Graph.* 37.4 (2018), 40:1–40:14 5.

[ZDZ*15] ZHANG, WANGYU, DENG, BAILIN, ZHANG, JUYONG, et al. "Guided Mesh Normal Filtering". *Comput. Graph. Forum* 34.7 (2015), 23–34 3.

[ZFAT11] ZHENG, YOUYI, FU, HONGBO, AU, OSCAR KIN-CHUNG, and TAI, CHIEW-LAN. "Bilateral Normal Filtering for Mesh Denoising". *IEEE Trans. Vis. Comput. Graph.* 17.10 (2011), 1521–1530 3.

[ZG16] ZHAO, HUI and GORTLER, STEVEN J. "A Report on Shape Deformation with a Stretching and Bending Energy". *CoRR* abs/1603.06821 (2016). arXiv: 1603.06821 5.

[ZHW*06] ZHOU, KUN, HUANG, XIN, WANG, XI, et al. "Mesh quilting for geometric texture synthesis". *ACM Trans. Graph.* 25.3 (2006), 690–697 2.

[ZJA21] ZHANG, JIAYI ERIS, JACOBSON, ALEC, and ALEXA, MARC. "Fast Updates for Least-Squares Rotational Alignment". *Computer Graphics Forum* (2021) 6.

[ZLL*17] ZHAO, HUI, LEI, NA, LI, XUAN, et al. "Robust Edge-Preserved Surface Mesh Polycube Deformation". *25th Pacific Conference on Computer Graphics and Applications, PG 2017 - Short Papers, Taipei, Taiwan, October 16-19, 2017*. Ed. by BARBIC, JERNEJ, LIN, WEN-CHIEH, and SORKINE-HORNUNG, OLGA. Eurographics Association, 2017, 17–22 2, 7.

[ZNI*14] ZOLLHÖFER, MICHAEL, NIESSNER, MATTHIAS, IZADI, SHAHRAM, et al. "Real-time non-rigid reconstruction using an RGB-D camera". *ACM Trans. Graph.* 33.4 (2014), 156:1–156:12 3.

[ZSL*20] ZHAO, HUI, SU, KEHUA, LI, CHENCHEN, et al. "Mesh Parametrization Driven by Unit Normal Flow". *Comput. Graph. Forum* 39.1 (2020), 34–49 3.

[ZWZD15] ZHANG, HUAYAN, WU, CHUNLIN, ZHANG, JUYONG, and DENG, JIANSONG. "Variational Mesh Denoising Using Total Variation and Piecewise Constant Function Space". *IEEE Trans. Vis. Comput. Graph.* 21.7 (2015), 873–886 3.

Appendix A: Local Step with E_{ARAP}

Given a fixed V', we obtain the optimal rotation for each vertex k by solving the following minimization problem

$$\mathsf{R}_k = \underset{\mathsf{R}_k \in SO(3)}{\arg\min} \sum_{i,j \in \mathcal{N}_k} w_{ij} \|\mathsf{R}_k \mathsf{e}_{ij} - \mathsf{e}'_{ij}\|_2^2 + \lambda a_k \|\mathsf{R}_k \hat{\mathsf{n}}_k - \mathsf{t}_k\|_2^2$$

The above optimization is an instance of the *orthogonal Procrustes* which finds the best rotation matrix R_k to map a set of vectors $(\mathsf{e}_{ij}, \hat{\mathsf{n}}_k)$ to another set of vectors $(\mathsf{e}'_{ij}, \mathsf{t}_k)$. We can re-write it into a more compact expression as:

$$\mathsf{R}_k^\star = \underset{\mathsf{R}_k \in SO(3)}{\arg\max} \ \operatorname{Tr}(\mathsf{R}_k \mathsf{X}_k) \tag{7}$$

$$\mathsf{X}_k = \begin{bmatrix} \mathsf{E}_k & \hat{\mathsf{n}}_k \end{bmatrix} \begin{bmatrix} \mathsf{W}_k & \\ & \lambda a_k \end{bmatrix} \begin{bmatrix} \mathsf{E}'^{\top}_k \\ \mathsf{t}_k^\top \end{bmatrix}. \tag{8}$$

where W_k is a $|\mathcal{N}_k|$-by-$|\mathcal{N}_k|$ diagonal matrix of the cotangent weights w_{ij}, E_k and E'_k are 3-by-$|\mathcal{N}_k|$ matrices concatenating the edge vectors of the face one-ring at the rest and deformed states, respectively. One can then derive the optimal R_k from the SVD of $\mathsf{X}_k = \mathcal{U}_k \Sigma_k \mathcal{V}_k^\top$

$$\mathsf{R}_k = \mathcal{V}_k \mathcal{U}_k^\top, \tag{9}$$

up to changing the sign of the column of $\mathcal{U}_k$ so that $\det(\mathsf{R}_k) > 0$.

Appendix B: Global Step with E_{ARAP}

The global step updates the deformed vertex positions V' from a fixed set of rotations R obtained via the local step. This boils down to solving the following problem

$$\mathsf{V}'^{\star} = \operatorname*{arg\,min}_{\mathsf{V}'} \sum_{k \in V} \sum_{i,j \in \mathcal{N}_k} w_{ij} \|\mathsf{R}_k \mathsf{e}_{ij} - \mathsf{e}'_{ij}\|_2^2$$

We can expand this energy as

$$\begin{aligned}
&\sum_{k \in V} \sum_{i,j \in \mathcal{N}_k} w_{ij} \|\mathsf{R}_k \mathsf{e}_{ij} - \mathsf{e}'_{ij}\|_2^2 \\
&\quad = \sum_{k \in V} \sum_{i,j \in f_k} w_{ij} \mathsf{e}'^{\top}_{ij} \mathsf{e}'_{ij} - 2 w_{ij} \mathsf{e}'^{\top}_{ij} \mathsf{R}_k \mathsf{e}_{ij} + \text{constant}
\end{aligned}$$

It is often convenient to express the summation in terms of matrices. We introduce a directed incidence matrix A_k with size $|V|$-by-$|\mathcal{N}_k|$ to represent the edge vectors in $\mathcal{N}_k$ as $\mathsf{V}^{\top}\mathsf{A}_k$, and we use M_k to represent a $|\mathcal{N}_k|$-by-$|\mathcal{N}_k|$ diagonal matrix of the weights w_{ij}. Then we can re-write the energy in terms of matrices as

$$\begin{aligned}
&\sum_{k \in V} \operatorname{Tr}(\mathsf{M}_k \mathsf{A}_k^{\top} \mathsf{V}' \mathsf{V}'^{\top} \mathsf{A}_k) - 2 \operatorname{Tr}(\mathsf{M}_k \mathsf{A}_k^{\top} \mathsf{V}' \mathsf{R}_k \mathsf{V}^{\top} \mathsf{A}_k) \\
&= \sum_{k \in V} \operatorname{Tr}(\mathsf{V}'^{\top} \mathsf{A}_k \mathsf{M}_k \mathsf{A}_k^{\top} \mathsf{V}') - 2 \operatorname{Tr}(\mathsf{R}_k \mathsf{V}^{\top} \mathsf{A}_k \mathsf{M}_k \mathsf{A}_k^{\top} \mathsf{V}') \\
&= \operatorname{Tr}\left(\mathsf{V}'^{\top} \left(\textstyle\sum_k \mathsf{A}_k \mathsf{M}_k \mathsf{A}_k^{\top}\right) \mathsf{V}'\right) - 2 \operatorname{Tr}\left(\left(\textstyle\sum_k \mathsf{R}_k \mathsf{V}^{\top} \mathsf{A}_k \mathsf{M}_k \mathsf{A}_k^{\top}\right) \mathsf{V}'\right) \\
&= \operatorname{Tr}(\mathsf{V}'^{\top} \mathsf{Q} \mathsf{V}') - 2 \operatorname{Tr}(\mathsf{R}\mathsf{K}\mathsf{V}'), \qquad (10)
\end{aligned}$$

where $\mathsf{R} = \{\mathsf{R}_k\}$ is the concatenation of all the rotations, Q is a $|V|$-by-$|V|$ symmetric matrix, and K is a $|9V|$-by-$|3V|$ matrix stacking the constant terms which can be computed during the precomputation. We can then find the optimal V' by solving a linear system

$$\mathsf{Q}\mathsf{V}' = \mathsf{K}^{\top}\mathsf{R}^{\top}$$

As we know from [SA07], Q is the cotangent Laplacian [PP93]. We can pre-factorize Q to speed up runtime performance. With these pieces in hand, we can minimize our energy Eq. 3 by iteratively performing the local and the global steps (see Alg. 1).

Appendix C: Generalize to E_{FARAP} and E_{ACAP}

Changing the regularization from E_{ARAP} to the membrane-only regularization E_{FARAP} (Eq. 4) requires to re-define R on each face and change the set of edge vectors to the three edge vectors of a triangle. These changes would lead us to replace the $\mathsf{E}_k, \mathsf{E}'_k$ in the local step Eq. 7 to the three edge vectors of a face, and a_k to the face area. In the global step, one only needs to update the incidence matrix A_k in Eq. 10 to a $|V|$-by-$|f_k|$ matrix containing the three edge vectors information.

Deploying the as-conformal-as-possible regularization E_{ACAP} (Eq. 5) changes the local step to solve an instance of the isotropic orthogonal Procrustes problem, where an analytical solution has been derived in [SC70]. In short, one can obtain the optimal rotation the same way as Eq. 9, and compute the optimal scaling s_k analytically as

$$s_k = \frac{\operatorname{Tr}(\mathsf{W}_k \mathsf{E}'^{\top}_k \mathsf{R}_k \mathsf{E}_k) + \lambda a_k \hat{\mathsf{n}}_k^{\top} \mathsf{t}_k}{\operatorname{Tr}(\mathsf{W}_k \mathsf{E}_k^{\top} \mathsf{E}_k) + \lambda a_k \hat{\mathsf{n}}_k^{\top} \cancel{\mathsf{t}_k}\, \hat{n}_k}.$$

When assembling the matrices for the global step, using E_{ACAP} would require replacing R_k with $s_k \mathsf{R}_k$.

Appendix D: Projective Dynamics for Dynamic Target Normals

We draw inspiration from *projective dynamics* [BML*14] to handle cases where the target normal T is a function of output geometry B'. Let us first define

$$T = \operatorname{arg\,min} E_N(\mathsf{V}')$$

as a minimizer of an energy E_N defined on the output shape. In our cases, E_N could be the distance to the closet normals or the developable energy [SGC18]. With this definition, we re-write Eq. 3 as

$$\begin{aligned}
&\min_{\mathsf{V}',\mathsf{R}} \sum_{k \in V} \sum_{i,j \in \mathcal{N}_k} w_{ij} \|\mathsf{R}_k \mathsf{e}_{ij} - \mathsf{e}'_{ij}\|_2^2 + \lambda a_k \|\mathsf{R}_k \hat{\mathsf{n}}_k - \mathsf{t}_k\|_2^2, \\
&\text{subject to } T = \operatorname{arg\,min} E_N(\mathsf{V}')
\end{aligned}$$

This reformulation allows us to directly deploy the projective dynamics solver by first projecting $T = \{\mathsf{t}_k\}$ to the "constraint" E_N, fixing t_k, and solving the original problem as Eq. 3 via the local/global solver to get V' at the next iteration. We then iterate this procedure (see Alg. 1) until convergence. This expression enables us to plug-and-play different E_N for different modeling objectives.

Appendix E: Normal Driven Developable Surfaces

Our normal-driven editing can be used to create developable surfaces by specifying a set of target normals that are developable. Stein et al. [SGC18] propose a characterization of discrete developability based on face normals of a vertex one-ring. In short, if all the one-ring face normals correspond to a common plane or two planes, then this local one-ring is piecewise developable.

With this characterization, we can easily get a set of "developable" face normals by (1) visiting all the one-ring faces of a vertex, (2) performing a small principle component analysis on the face normals for each one-ring, and (3) projecting the normals to one or two common planes by zeroing out the components correspond to the smallest eigenvalues. By using a different threshold to decide whether to zero out the smallest or the smallest two components, we can control the amount of creases in the developable approximation (see Fig. 19). As each face will receive three (possibly) different developable normals from the previous procedure, we simply average them to get the target face normals. We perform this developable normal computation at each iteration in parallel, which corresponds to the Line 9 of Alg. 1.